ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA087

# THE TIMING DIAGRAM EDITING AND VERIFICATION METHOD

G. Fatkin*, A. Senchenko, BINP SB RAS and NSU, Novosibirsk, Russia

*Abstract*

Preparation and verification of the timing diagrams for the modern complex facilities with diversified timing systems is a difficult task. A mathematical method for convenient editing and verification of the timing diagrams is presented. This method is based on systems of linear equations and linear inequalities. Every timing diagram has three interconnected representations: a textual equation representation, a matrix representation and a graph (tree) representation. A prototype of the software using this method was conceived in Python. This prototype allows conversion of the timing data between all three representations and its visualization.

## INTRODUCTION

Nowadays large experimental facilities often have a lot of devices and subsystems which require careful orchestration and synchronisation. Therefore configuration of timing subsystem often requires elaborate formation of the time diagrams. These time diagrams are mostly compiled and checked manually. Such procedure, being a difficult task by itself, becomes even more complex when several subsystems are interrelated or specific timing requirements have to be met. We propose a method for describing, compiling and storing time diagrams, checking their consistency and setting the relevant delay values in hardware.

## FORMALISM

Time diagram is essentially the relationship between the times of different events on the installation. Let us assume for the sake of simplicity that all the clocks are synchronized. If it is not the case, the time skews could easily be taken into account by adding some additional equations. We will also reduce all the units to the common order and omit the dimension, e.g. we could count all the times in ns. The relationship between the time of two events could be then described by the following linear equation:

$$t_2 = t_1 + b, \tag{1}$$

where $t_2$ and $t_1$ – times of the events, and $b$ – is the delay. The value of $b$ can be positive, specifying that the second event happens after the first, negative meaning that the second event happens before the first, or zero, indicating that events are simultaneous.

Let us consider a time diagram of a traffic light as an example. It consists of three events: at time moment 0 the green signal is lit, after 5 seconds, the yellow is ignited and 5 seconds later the red signal is lit. Let us designate these events as $t_{green}$, $t_{yellow}$, $t_{red}$. Their interrelation is described by the following system:

$$\begin{cases} t_{green} = 0 \\ t_{yellow} = t_{green} + 5 \\ t_{red} = t_{yellow} + 5. \end{cases} \tag{2}$$

Using matrix notation, it could be written as:

$$\begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ 5 \end{pmatrix} \tag{3}$$

Let us note that we have numerated the events, threrefore we have to keep their corresponding names in a vector:

$$v = \begin{pmatrix} ''Green'' \\ ''Yellow'' \\ ''Red'' \end{pmatrix} \tag{4}$$

In general case, the time diagram of an installation could be described by a system of linear equations which could be written in the following form:

$$At = b, \tag{5}$$

where t - the vector of the event times, $A$ - the system matrix, and $b$ - the delay vector. We also need a vector $v$ of the event names.

For the time diagram to be correct, the matrix $A$ has to be invertible, in this case the system 5 is solvable. If matrix $A$ is not invertible, then there was an error in time diagram specification. Let us formulate several other properties of the matrix $A$:

1. Diagonal elements are 1.
2. Other elemets are either 0, or -1.
3. For a large system $A$ must be a sparse matrix (most of its elements are 0).

Last property is rather important in practice, because it allows to choose an effective way of the matrix storage and linear equations system solution.

## EVENTS GRAPH

The matrix representation of time diagram is quite convenient for storage and computation of the event times but it doesn't allow to visually explore the relationship of events. Let us deduce the following matrix:

$$G = E - A, \tag{6}$$

where $E$ - is an eye matrix of the same dimension as $A$.

We can note that the matrix $G$ is an adjacency matrix of a directed graph. The vertex of a graph represents an event

* G.A.Fatkin@inp.nsk.su

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA087



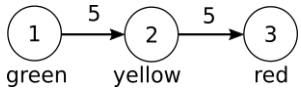


Figure 1: Graph of the events for the traffic light.

and an edge represents a relationship. It is logical to place a name of the event in the vertex, and to make the weight of an edge equal to the appropriate element of the vector B. Thus we get a representation of time diagram as a graph.

For the example let us construct the graph for a traffic light.

$$G = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{7}$$

This graph is presented at fig. 1. For the inverse transformation of the graph to the system of linear equations we should calculate:

$$A = E - G, \tag{8}$$

and vector $b$ should be the appropriate weights of the graph edges.

It is easy to show, that because of the properties of the matrix $A$, this graph is a directed graph without cycles, such graphs are called trees. We could identify several trees by searching for all the connected nodes. That way we can probably identify subsystems. That will also help us to determine a casual relationship between events. We are also planning to use this tree representation for the tree view control and therefore ease the operator's work.

## TIME CONSTRAINTS

It is usual to impose some constraints on the time diagram. Such constraints could be on the relative position of the events, total diagram duration, positioning event in some range, etc. All these constraints could be represented as one or several linear inequalities. Let us assume for example that event $t_1$ should happen in the interval $[3, 6]$, and event $t_2$ should be later than event $t_1$ from 2 to 5 time units. This could be represented as the following system of linear inequalities:

$$\begin{cases} 3 \le t_1 \le 6 \\ 2 \le t_2 - t_1 \le 5 \end{cases} \tag{9}$$

For the sake of simplicity we will analyze only non-strict inequalites, because in practice any strict inequality could be presented as a non-strict one with slightly moved limit:

$$a < b \leftrightarrow a \le b - \delta, \delta << 1. \tag{10}$$

We can also notice that the sign of the inequality could easily be changed multiplying left and right parts by $-1$:

$$a \le b \leftrightarrow -a \ge -b. \tag{11}$$

Using this property we can make all the inequalities use the same sign. Therefore all the time constraints could be presented as the following system of linear inequalitis:

$$Ct \ge d, \tag{12}$$

where $C$ - matrix of inequalities, $t$ - event times vector, and $d$ - constraints vector.

E.g. (9), will have the following form:

$$\begin{pmatrix} 1 & 0 \\ -1 & 0 \\ -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \ge \begin{pmatrix} 3 \\ -6 \\ 2 \\ -5 \end{pmatrix} \tag{13}$$

We can notice that for the large system matrix $C$ would be sparced ( as the matrix $A$) and would be composed mosly of 0, 1 and -1. If we have the matrix $C$ and the event times vector $t$ it is very easy to check whether this vector satisfies the inequality. We can count the value of the vector $Ct$, and check if all the components ar greater than $d$.

But there is a probability of an error when we specify the inequalities system. Therefore we have to know whether the system is solvable, namely if there is any vector $t$, so that $Ct \ge d$.

Let us present the system in a canonical form. For that let us introduce the vector of the lag variables $w$,thus $w = Ct - d$. Then the equivalent problem is formulated as follows:

$$(E - A) \begin{pmatrix} w \\ x \end{pmatrix} = b, w \ge 0. \tag{14}$$

This problem is reduced to the classical problem of a linear programming being the problem of minimization of linear functional with some boundary conditions:

$$min\{(c, y), Dy = e, y \ge 0\}, \tag{15}$$

where we have to assume the objective function c = 0. To solve this problem we can use an algorythm using the simplex-method (e.g. linprog from scipy.optimize [1, 2]). The algorythm will eithr provide us the first vertex of the convex polyhedron which is the edge of the solutions of the system of linear inequalities, or it will determine that the system is unsolvable.

## PRACTICAL REALIZATION

For the prototype practical realization a Python program using numpy [3], scipy and networkx [4] was developed. The program consists of several parts: Time Editor, Solver, Export module. Time editor allows to construct matrix $A$ and vector set $b, v$ by parsing a set of lines in the following form:

```
Start = 0
HVCharge = Start + 50 us
ArcIgnition = Start + 30 ms
Adc0 = ArcIgnition
Degauss = ArcIgnition + 70 ms + 5800 us
ModulatorA = Degauss + 200 us - 45 ns
ModulatorB = Degauss + 200 us - 25 ns
```

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA087

```
ModulatorC = Degauss + 200 us − 15 ns
ModulatorD = Degauss + 200 us − 10 ns
ModulatorE = Degauss + 200 us + 10 ns
ModulatorF = Degauss + 200 us + 45 ns
ModulatorG = Degauss + 200 us + 15 ns
ModulatorH = Degauss + 200 us − 30 ns
```

A parser is capable to convert dimensions, the base unit is chosen to be 1 ns. The equations editor based on Scintilla component is provided. The program also allows to create a list of lines from the system of linear equations. The conversion is not reversible, as all the coefficients in the matrix form are added (consider line 5 and further on). An example of the unifying GUI is shown on fig. 2.

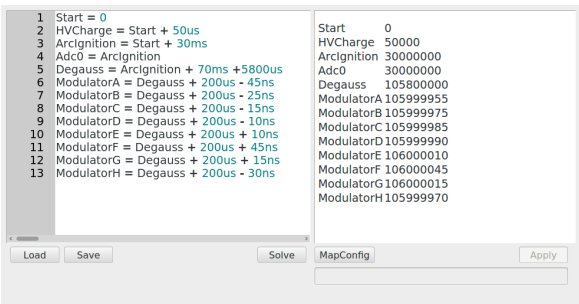


Figure 2: Example of a program window.

To store and create matrix $A$, a COO (COOrdinate) format is used. In the Solver it is converted to CSR (Compressed sparse row) using scipy. The Solver solves the linear equations system using scipy, or determines that the matrix is irreversible. The conversion to graph and plotting the graph is also realized. An example of the resulting plot is shown on fig. 3.

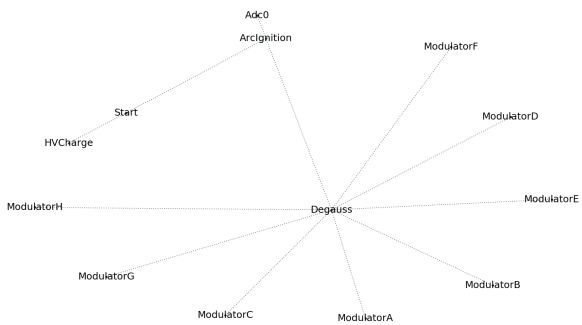


Figure 3: Events graph.

After the solution is recieved, we have to export the resulting times vector to the control system. For this task, a separate program accepting a vector and channel names in JSON format was written in Python. Based on mapping rules it transfers times vector values to the appropriate devices in used control system software. Currently we have a TANGO implementation written in pyTango for our timing subsystem. It is schematically shown on fig. 4.

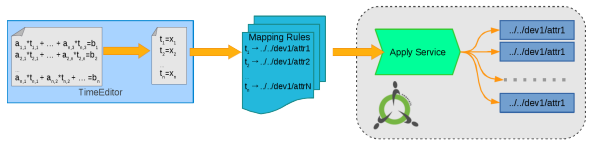


Figure 4: Time editor export to TANGO scheme

## CONCLUSION

A method was formulated for editing the time diagrams of the complex physical facilities. A prototype realization of editor and solver was written in Python, and an export module for TANGO was prototyped. We are planning to test it in LIA-20 control system [5]. We also plan to create an export module from the solver for EPICS and CX-Server control systems and use it to create time diagrams on functioning BINP accelerators.

We also plan to realize the check on the diagram consistency using a system of linear inequalities. The ideal evolution of the program is the creation of an interactive instrument that will allow: editing time diagrams in any of the three forms (text representation, matrices or graph) and plotting the events line of the time diagram.

## REFERENCES

[1] George B. Dantzig, *Linear programming and extensions*, Rand Corporation Research Study Princeton Univ. Press, Princeton, NJ, USA, 1963.

[2] S.H. Hillier and G.J. Lieberman, *Introduction to Mathematical Programming*, McGraw-Hill, 1995, Chapter 4.

[3] S. van der Walt, S.C. Colbert, and G. Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation", *Computing in Science & Engineering*, vol. 13, pp. 22–30, 2011.

[4] A.A. Hagberg, D.A. Schult, and P.J. Swart, "Exploring network structure, dynamics, and function using NetworkX", in *Proc. SciPy 2008*, Pasadena, CA, USA, Aug, 2008, pp. 11–15.

[5] G.A. Fatkin *et al.*, "LIA-20 Control System Project", presented at ICALEPCS'17, Barcelona, Spain, Oct. 2017, paper TH-PHA052.